

Université de Montréal
École des Hautes Études Commerciales
École Polytechnique

Université de Montréal
Centre de recherche sur les transports
3150 Avenue du Centre-ville
Montréal, Québec H3T 1J6
Téléphone: (514) 343-4343
Télécopieur: (514) 343-4347
Fax: (514) 343-4347
E-mail: crts@crts.hec.ca

CRT-2000-26



Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

***A CONSTRAINT PROGRAMMING LOCAL SEARCH
ALGORITHM FOR PHYSICIAN SCHEDULING***

by

Gianluca Cingini

June 2000

**PUBLICATION
CRT-2000-26**

***A CONSTRAINT PROGRAMMING LOCAL SEARCH
ALGORITHM FOR PHYSICIAN SCHEDULING****

by

Gianluca Cangini¹

¹ Centre de recherche sur les transports, Université de Montréal and Università degli Studi di Bologna,
Facoltà di Ingegneria, D.E.I.S., Viale Risorgimento 2, Bologna ITALY

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2000

Abstract

What we present here is an application of Constraint Programming for Physician Scheduling. The goal is not to find a solution to the problem by constructing a feasible table of work assignment, but to quickly optimize a solution we already have. For this purpose, a Tabu Search approach is proposed, but always based on constraint programming. Real data from the Côte-des-Neiges Hospital in Montréal shows how optimization is not an easy task to be accomplished, mostly because of the several parameters to be taken into account in evaluation.

Key words: Constraint Programming, Local Search, Tabu Search, Physician Scheduling, Optimization.

Résumé

Nous présentons ici une application de Programmation par contraintes pour un problème de planification d'horaires de médecins. Le but n'est pas de générer un emploi du temps pour les médecins, mais de rapidement en optimiser un déjà connu. Pour ce faire, on se propose d'utiliser la Recherche avec tabous, mais toujours dans le cadre de la programmation par contraintes. Les données réelles de l'Hôpital Côte-des-Neiges de Montréal nous montrent à quel point l'optimisation n'est pas une tâche aisée, surtout en raison des nombreux paramètres à considérer pour l'évaluation.

Mots clés : Programmation par Contraintes, Recherche Locale, Recherche avec Tabous, Problèmes d'allocation de personnel, Optimisation.

Introduction

Scheduling is nowadays a really lively field of research. Many different techniques and algorithms are developed, applied and tested, with mixed results, each new technique showing some new useful feature or improvement, but maybe some drawbacks too. Also combinations of different techniques have already been proposed.

Starting from the interesting framework proposed by [PG99] which integrates constraint programming and local search methods, our algorithm follows that integration, trying to reach a good and useful interaction of such techniques. Constraint programming, with its intuitive simplicity and powerful constraint propagation, makes the code 'light', easy to write, use, maintain and modify. Local Search offers a much more efficient search through solutions if compared with branch-and-bound algorithms. Tabu Search increases the efficiency of local search in trying to find a global optimum. A good integration of these techniques takes advantage of the strength of each of them and the final result is an algorithm very powerful and very simple at the same time. [Tri98] finds quickly good solutions but which can still be improved. We apply here a post-optimization to such solutions.

In section 1 we present the main problem of human resource management and our particular physician scheduling problem, with the constraints given by a real hospital. In section 2 we explain in detail the mathematical model we chose for our problem. In section 3 we present our approach to the problem and how we intend to use and apply constraint programming, local search and tabu search. Section 4 offers more detailed information about our implementation's choices and solutions. Without descending too deep into the actual code, it gives a good idea of how the program is structured. In section 5 the results of several tests are shown and commented, while more extended tables of results are shown in the Appendix.

Note: the references to physicians in the text are always in the masculine form simply to avoid tedious he/she, his/her notations.

1 Problem

1.1 General Problem : Human Resource Management

The general human resource management problem consists of placing employees in work shifts over a certain period of time. It may be viewed as a resource allocation problem: we have employees as resources and work shifts as tasks. The number of employees, shifts per day and days in the plan vary from one case to another. The goal is to fill all the work shifts of the scheduling horizon with one or more employees. What makes this problem challenging and difficult to solve manually or automatically is not only the large number of employees or shifts, but all the constraints that may exist, which reduce dramatically the number of acceptable solutions.

A generic typical example of such constraints could be:

- every employee must fill at most one work shift per day
- employees cannot work consecutive nights
- every employee must have a day off every week and/or a week off every n months
- there must be at least one weekend off between two weekend work shifts of an employee
- the total number of shifts per week of each employee must not exceed a certain quota

And also very particular constraints may exist, such as:

- employee x cannot work during mornings
- employee x and y do not want to work together
- employee x prefers working on saturday afternoons
- employee x must work more than all the others in this period, since he had more holiday in the previous period

It is necessary to keep separate the different types of constraints. What we can call the *hard constraints* of the problem are those which really give the structure of the problem and which cannot be violated, otherwise the

solution is not feasible. On the other hand, we can call *soft constraints* those which lead to a better or a worst solution (if respected or not) according to a certain cost function, but which can eventually be violated without making the solution infeasible.

1.2 Particular Problem : Physician Scheduling

In our particular case the employees are physicians who must fill the work shifts of a hospital's geriatrics ward. The obvious goal is that every day, at every time, there should be a physician in the ward, or at least there should be one available on call during nights. We used real data from the Côte-des-Neiges Hospital in Montréal which consist in fifteen physicians to be scheduled over a period of three months with three shifts per day: morning (8am to 1pm), afternoon (1pm to 6pm) and night (6pm to 8am). The hospital also sets all the constraints to be respected to produce the timetable.

Hard constraints

- During *Holidays* and *Weekends*:
 - The physician in charge for that day will cover all the three shifts (morning, afternoon and night).
 - A physician can be in charge two consecutive days, such as saturday and sunday.
- During *Regular Days* (i.e. not Holidays or Weekends):
 - A physician cannot be assigned to the morning and afternoon shifts of the same day
 - No consecutive night shifts are given to a physician
 - A physician cannot be assigned a night shift and the next morning shift
 - Every physician can indicate the shifts for which he's not available throughout the whole period.

Soft Constraints

- Every physician has a status which consists of the number of hours per week he's supposed to work (35h corresponding to full time). Hence the workload assigned to each physician should be proportional to that status.

- The workload for each physician should also take into account his availability over the current period and his actual workload from previous scheduling horizons (to have a good global workload balance between physicians at the end of the year).
- On the other hand, the workload during Holidays and Weekends should be equally distributed between all physicians, regardless of the status or other factors.
- A physician should have a balanced workload throughout the period. His shifts should be spread out as much as possible, that is to say for example that a full-time physician with a target workload of fourteen shifts over fourteen weeks should work one day per week and not several times during certain weeks and never during others.
- Every physician can submit his list of preferred shifts over the period, and if he prefers to be assigned weekends in block (saturday and sunday of the same week), or separated.

As we already said, all those soft constraints lead to a good solution if respected, but the solution remains a valid one if they are not respected.

2 Model

The two principal models for this kind of problem, from the literature, are the "*Shift-Centered Model*" and the "*Physician-Centered Model*". The former consists of one decision variable for every slot in the schedule, which can assume a value from the set of physicians. The latter, on the other hand, has several variables per physician. Those variables should indicate, in chronological order, the list of all the work shifts assigned to that physician. It is possible to have for every physician the whole list of work shifts with values true or false if the shifts are assigned to that physician or not. In this case we have to use a great amount of memory even if a physician is assigned only to few shifts. Otherwise we can have for every physician only the exact list of work shifts he is assigned to, but we must know in advance how many shifts are assigned to each physician. The physician-centered model requires that we match exactly the computed individual workloads, which may involve trial and error in some contexts. The shift-centered model is more flexible and we

quickly get a fair solution. Even though the physician-centered model with a two-phase approach seemed to give the best results (see [Pes00]), here we use the shift-centered model. A first reason for this choice could be that it is more 'natural' and intuitive, and so simpler to comprehend and use. But the main reason is that the shift-centered model appears the most appropriate for the application of a Local Search Method, identifying immediately a move of the local search with the substitution of the value of a slot with another physician (see section 3.2). With the physician-centered model the move should delete a shift from a physician's list and insert it, in the correct chronological order, in another physician's list. It results then to be a much more complex operation, which involves several variables and it requires changing the structure of the model.

Then, following our choice, the shift-centered model will be mathematically defined like this:

- set of weeks: $W = \{1..n\}$
where n is the number of weeks in the scheduling horizon
- set of days: $D = \{Mon, Tue, Wed, Thu, Fri, Sat, Sun\}$
Sat and Sun: Weekends, and Mon to Fri: Regular Days
- set of shifts per day: $S = \{am, pm, night\}$
- set of physicians: P
- every single shift: $\langle w, i, j \rangle$ with $w \in W, i \in D, j \in S$
- the table of shifts: $T = \{\langle w, i, j \rangle : w \in W, i \in D, j \in S\}$
in which every slot corresponds to a decision variable:

$$s_{wij} \in P, \langle w, i, j \rangle \in T$$

If we define the useful sets: S_h as the set of *Holidays* shifts, and U_k as the set of shifts for which the physician k is unavailable, the constraints of the problem will be:

$$s_{wij} = s_{wij'} \quad \forall w \in W, \forall i \in Weekends, \forall j, j' \in S \quad (1)$$

$$s_{wij} = s_{wij'} \quad \forall \langle w, i, j \rangle, \langle w, i, j' \rangle \in S_h \quad (2)$$

$$s_{w \ i \ night} \neq s_{w \ i+1 \ night} \quad \forall w \in W, i \in \{Mon..Thu\} \quad (3)$$

$$s_{w\ i\ night} \neq s_{w\ i+1\ am} \quad \forall w \in W, \forall i \in RegularDays \quad (4)$$

$$s_{w\ Sun\ night} \neq s_{w\ Mon\ am} \quad \forall w \in W \setminus \{n\} \quad (5)$$

$$s_{w\ i\ am} \neq s_{w\ i\ pm} \quad \forall w \in W, \forall i \in RegularDays, \langle w, i, am \rangle \notin S_h \quad (6)$$

$$s_{w\ i\ j} \neq k \quad \forall k \in P, \forall \langle w, i, j \rangle \in U_k \quad (7)$$

Which represent:

1. all three shifts (*am*, *pm*, *night*) during Weekends
2. all three shifts (*am*, *pm*, *night*) during Holidays
3. no consecutive *night* shifts in Regular Days
4. no consecutive *night* and *am* shifts in Regular Days
5. no consecutive *Sun night* and *Mon am* shifts in Regular Days
6. no *am* and *pm* shifts on the same day on Regular Days
7. physicians availability

3 Approach

3.1 Constraint Programming

The nature of the problem itself calls for Constraint Programming as the most simple and immediate way to implement our model. In fact, to pose the constraints on the array representing the table of shifts requires no more than twenty lines of code thanks to constraint programming. As a reference to constraint programming we suggest [MS98]. The language used was ILOG Solver 4.4, which is a C++ library and it is very portable and easy to interconnect with other existing programs. Moreover, the powerful constraint programming provided by ILOG Solver library gives the algorithm great efficiency, while the modularity offered by C++ makes it highly reusable and relocable. In particular, the models of the main problem and of neighbourhoods ¹ are contained in different modules. This means that to change the constraints of the problem, i.e. to use the program for another hospital which has different requirements, it will be simply necessary to substitute the desired module.

¹See section 3.2 for the definition of "neighbourhood".

3.2 Local Search

Starting from a good initial solution (given from the work of [Tri98]), a Local Search Method was applied to optimize it. The local search method we used is an implementation of the framework proposed by [PG99].

The main idea of local search is to start from a feasible timetable, to make a move to pass to a new timetable and to iterate this process. A *move* consists of replacing the current physician in a slot with another one. We can both choose the slot first and then the physician, or the other way around. We obtain in that way a *neighbourhood* of solutions. This means a set of solutions which differs from the initial one only for the value of one single slot in the whole table. To choose from all possible moves we decided to follow a *best-improvement* criterion. In next implementations, a first-improvement search could be tried, to see if it will arrive in a faster computational time at the same level of optimization. Once the best possible move is found and the slot finally changed, the new table becomes the current solution and the process is then iterated until we find an optimal solution² or we decide the current solution is 'good enough'.

Our *neighbourhood model* is defined by the two decision variables:

$$p \in P$$

$$t \in T$$

And the constraint:

$$p \neq \sigma(t)$$

where $\sigma : T \rightarrow P$ represents the current solution.

Even the local search is implemented with constraint programming by triggering particular so called *interface constraints* when an assignment is made to one or both of our decision variables. This process is graphically explained in figure 1.

Defining $\langle a, b, c \rangle$ the slot and k the physician of the chosen local move, the interface constraints are the following³:

$$t = \langle a, b, c \rangle \Rightarrow \bigwedge_{(w,i,j) \in T \setminus \{\langle a,b,c \rangle\}} s_{wij} = \sigma(\langle w, i, j \rangle) \quad (8)$$

$$t = \langle a, b, c \rangle \Rightarrow p = s_{abc} \quad (9)$$

²The criteria to say if a solution is optimal are given in section 5.

³The numbers on the side are also referred to in the program's code.

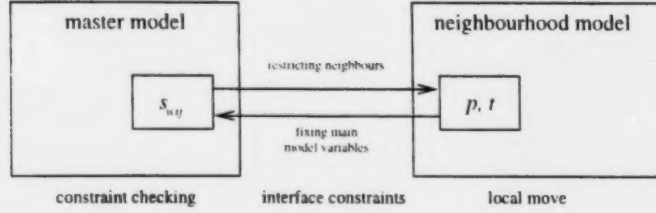


Figure 1: interaction between the master and neighbourhood models.

$$p = k \Rightarrow \bigwedge_{(w,i,j) \in T \wedge \sigma((w,i,j))=k} s_{wij} = k \quad (10)$$

$$p = k \Rightarrow \bigwedge_{(w,i,j) \in T \wedge k \notin \text{domain}(s_{wij})} s_{wij} = \sigma((w,i,j)) \quad (11)$$

$$p = k \Rightarrow t \in \{(w,i,j) \in T : k \in \text{domain}(s_{wij})\} \quad (12)$$

Which mean:

- once t is instantiated, we keep all the other slots of the table with their old values (8), and we restrict the domain of p to that of the chosen slot (9)
- once p is instantiated, we keep all the slots with already that value (10), as well as all those which do not have the chosen physician k in their domain (11). Moreover, the domain of t is restricted to only the slots which admit k as a possible value (12)

For more details about the framework, the reader is invited to refer to [PG99].

3.2.1 Neighbourhoods

Three different neighbourhoods were created for each of *Holidays*, *Weekends* and *Regular Days*.

The reason for creating three neighbourhoods is the difference existing in variable domains and/or in interface constraints. Variable p remains always the physician of our local move with domain P . On the other hand the domain of t is different in the three cases, i.e. a subset of T corresponding to holiday, weekend and regular day slots respectively. The interface constraints presented ((8)-(12)) are those of the regular days neighbourhood (the larger and more interesting). For weekends and holidays they would be slightly

different because of the constraint which imposes in that case to have the same physician in charge of the whole day, but the main idea remains the same. Our algorithm remains at any rate general, since the function that performs the tabu search is always the same. It is simply necessary to specify on which neighbourhood to apply it. This is also the reason why we applied local search on the holidays neighbourhood. In fact this is a really small neighbourhood (only two slots!) which doesn't really require any search at all, but only a single simple move which could be even made manually. But since we already had the data structures and functions necessary for the local search, we decided to maintain uniformity with the other neighbourhoods and apply the same optimization process⁴.

3.3 Tabu Search

The main problem of local search is that it could lead to a local optimum and never move from there, since it would not find any move which improves the evaluation. There are several methods to avoid this problem. Starting from different initial solutions, randomly choosing the move, Simulated Annealing, Tabu Search and Genetic Algorithms are just some of the algorithms proposed in the literature (see [Ree93]).

We decided to apply tabu search since it was both simple and efficient for our purposes.

Roughly speaking, tabu search is a technique which does not allow certain moves to be made, according to recency or frequency criteria. It is so to avoid circular search through solutions or to force 'uphill' moves to exit from a local optimum, as in our case. See figure 2 for a graphical explanation.

It is always possible to set the so called *aspiration criterion*, which determines if and when it is allowed for a tabu move to be overridden. For a more detailed description of tabu search, the reader could refer to [GL93]. We applied tabu search in its most 'classical' form: using a memory for the most recent moves, and allowing a tabu move to be overridden only in the case it leads to a better cost compared with the best one found so far.

⁴Moreover you never know where and when the algorithm could be used, and maybe in some country, in particular periods of the year, they could have a surprisingly larger number of holidays.

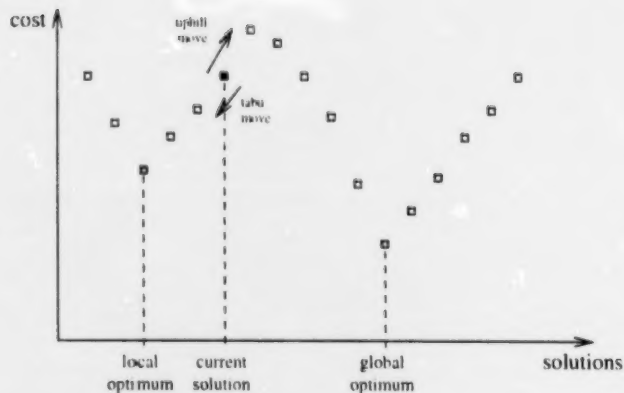


Figure 2: illustration of an uphill and tabu move

4 Implementation

As we already mentioned, the solution method is implemented using constraint programming and coded with the C++ library ILOG Solver 4.4. The master model of the problem (i.e. the hard constraints) is in the module “*master*”. The regular days neighbourhood is in the module “*neighRD*” while the weekend and holiday neighbourhoods are together in the module “*neighWH*”⁵. The initialization of all the global parameters of the problem, such as the dimension of the table and all information about physicians (total number, names, status, availabilities, preferences...) are read from files “*init.dat*”, “*data.dat*”, “*cumul.dat*”. The initial solution is read from the file “*sol.dat*”. Whatever of this file is missing or damaged, the program should be able to continue his task by producing a feasible initial solution or by initializing the global parameters with default values. The target workload for each physician is computed by considering all possible factors, such as physicians status, preferences, absences and cumulative work from previous periods (see also [Pes00]). The function which performs this task is contained in the module “*target*”. The main module “*optimize*” contains all the functions to apply the tabu search, and to evaluate solutions during the search. Finally, in the module “*utility*” are contained all global variable declarations

⁵We decided to do so because of the similarity of their interface constraints that makes them conceptually identical.

and all the useful functions such as, for example, those for input/output. The main data structures we used are:

- A tri-dimensional array ([week][day][day shift]) for the model and the current/best/bestEver solutions. Each slot of the array takes an integer value, corresponding to the number of the physician assigned to that work shift.
- A bi-dimensional array ([work shift][physician]) to store the tabu moves. Each slot $[i][j]$ corresponds to the move placing physician j in work shift i . The value assigned to each slot of the array is an integer which indicates the iteration until which the move is to be considered tabu. To make it clearer: every time we test a move $[i][j]$, we look if the number in the tabu array is greater than the current iteration. If it is so, the move is tabu. Otherwise it is possible to make the move and the tabu array is updated by adding as tabu the move $[i][j']$, where j' is the physician that is currently assigned to the work shift i . The move will be tabu for a random number of iterations, chosen uniformly between 6 and 11. This will avoid the possibility of returning to the same solution for that number of iterations.
- A *History* structure which contains the current solution and its cost, the current workload of each physician, the best solution found so far and its cost, the values of the neighbourhood model variables for the current best local move and the cost it will produce.

The tabu search is applied to the three different neighbourhoods of regular days, weekends and holidays (not necessarily in this order) and it stops when either:

1. an optimal solution is found
2. a certain number of consecutive uphill moves are made
3. no moves can be made for a certain consecutive number of iterations
4. the maximum number of iterations is reached

How to recognize a solution as optimal (1) is explained in section 5; but it is also shown that for a solution to have an optimal cost is only possible when the single factor to consider for the evaluation is the target workload.

In fact, when preferences and balance also have to be taken into account, the evaluation becomes much more complex and the optimal value is only an ideal, maybe not really reachable. The maximum number of uphill moves (2) and of iterations without moves (3) are set according to the size of the neighbourhood on which the local search is applied. These limits are set to avoid doing too many uphill moves which can lead us very far from the optimum, and to avoid continuing the search if no better solution can be found. The maximum number of iterations (4) is a parameter that can be fixed by the user ⁶ if he wants for some reason to put a limit to the search.

Once the CP model and tabu search were implemented, the most challenging part of the problem was to write good evaluation functions to make them work properly. By 'good evaluation function', we mean a function capable of returning a multi-criteria evaluation which considers the target workload of each physician, the preferences, and the balance of the workload over the period. For this purpose two parameters can be set by the user⁷: the bonus for the satisfied preferences, and the negative bonus for the not respected distances between two shifts of a physician (to have a well balanced workload). By setting these two parameters to zero, the evaluation function will only consider the target workload. Otherwise, preferences and balance will be considered, proportionally to the values assigned to those parameters.

5 Results

We tested the program with the real data from the Côte-des-Neiges Hospital in Montréal on a SUN Sparc Ultra 10 machine. The tests were made with values 0.5 as bonus for preference and 0.2 as negative bonus for not respected distances. We consider these to be reasonable mean values, which means that with higher values the program will consider much more preferences and/or balance than the target, and vice versa with lower values. The ideal cost of an optimal solution should be -90, which is calculated for the best possible case, by considering that all physician's workloads reach the target, with a perfect balance through the period and all preferences satisfied. It is not really possible to reach that optimum, since some preferences are in conflict with each other and/or with a good balance. Its only purpose is then to

⁶By setting the appropriate variable value in the file "*init.dat*".

⁷Again, these parameters are contained in the "*init.dat*" file, with all the other global parameters of the problem.

give us a qualitative idea of a good cost. The initial solution was taken from the work of [Tri98]. Its cost, with some additional information, is shown in table 1.

targets not reached	25
distances not respected	(12 day, 7 night, 1 weekend) 20
weekend preferences not resp.	0
satisfied preferences	83
satisfied weekend preferences	20
COST	18.9

Table 1: parameters of the initial solution

Before trying out our global optimization, we tested the tabu search on the three neighbourhoods separately to see if it worked properly on each of them. As a sort of validation of our algorithm, we also set the bonus variables both to zero, which means the evaluation only considers the target and not preferences and balance. In this case the optimal cost is zero and it is theoretically reachable. We obtained an optimal solution on every neighbourhood. The target of each physician was exactly reached and that proves the efficiency of our search. The test on the neighbourhoods with preferences and balance evaluation are shown on table 2, which indicate the number of moves and the time necessary, as well as the gain in the cost. The Regular Days neighbourhood requires more moves and more time since it is fairly larger than the others. In fact after the instantiation of the physician of our local move⁸ and the subsequent propagation of constraints, the domain of the slot has a mean cardinality of 90. This means that if we should use a branch-and-bound algorithm to try all the possible moves, our search tree would have about 1350 leaves (15 physicians x 90 slots).

Finally we combined the search on all three neighbourhoods. We obtained the best global result by applying tabu search on neighbourhoods in this order: Holidays, Weekends and Regular Days. Evaluation parameters of the final solution are shown in table 3. They clearly show that we improved the scheduling by reaching more targets. At the same time the workload balance for night shifts was improved and almost all the satisfied preferences

⁸The physician variable is the first to be instantiated since it has a smaller domain compared to that one of the slot variable.

Neighbourhood	Moves	Cost Improvement	Time (sec.)
Holidays	2	4.0	0.04
Weekends	7	13.5	0.62
Regular Days	16	37.1	4.12

Table 2: neighbourhoods tests results

maintained. Only 37 moves and 5.03 seconds of computing time were necessary. If interested, the reader can find much more detailed result tables in the Appendix.

targets not reached	2
distances not respected	(14 day, 1 night, 1 weekend) 16
weekend preferences not resp.	0
satisfied preferences	81
satisfied weekend preferences	19
COST	- 35.2

Table 3: parameters of the final solution

As last test, just to have another measurement of the efficiency of the algorithm, we tried out the optimization of a very bad initial solution. Its starting cost was **8916.8**, with no targets, distances and preferences respected. About 319 moves and almost 80.72 seconds computing were necessary to reach a good result, shown in table 4.

not reached targets	0
not respected distances	(9 day, 0 night, 0 weekend) 9
not resp. weekend preferences	2
satisfied preferences	50
satisfied weekend preferences	18
COST	- 28.8

Table 4: parameters of the final solution, starting from a totally bad initial solution

The final target and the balance parameters are very good, but not so many preferences are satisfied. This shows more clearly that it is important to have a good initial solution. On the other hand it shows also that our algorithm works in the right direction. So once we have a good solution, it could be wiser to use the post-optimization algorithm we propose instead of spending time and energy in trying to find a completely new solution, hoping it will be a better one.

Conclusion

The Constraint Programming Framework proposed by [PG99] was successfully implemented and the results show that once applied to a good initial solution, it leads very easily and quite quickly to an optimal solution, or at least to a much better one.

Among all the possible improvements to the program, what we propose could be:

- to consider the possibility of making two local moves at the same time (i.e. choosing two slots to be changed).
- to try out domain splitting, for cases of large neighbourhood such as *Regular Days* in order to speed up neighbourhood explorations
- to try a first-improvement search.
- to experiment on a greater number of initial solutions.

References

- [GL93] Fred Glover and Manuel Laguna. Tabu search. In *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–150. Halsted Press, 1993.
- [ILO96a] ILOG. *ILOG Solver 3.2 - Reference Manual*, 1996.
- [ILO96b] ILOG. *ILOG Solver 3.2 - User Manual*, 1996.
- [JSV96] B. Jaumard, F. Semet, and T. Vovor. A generalized linear programming model for nurse scheduling. *Groupe d'études et de recherche en analyse des décisions (G.E.R.A.D.)*, Cahier n.35, 1996.
- [MS98] Kim Marriott and Peter J. Stuckey. *Programming with constraints: an introduction*. MIT Press, 1998.
- [Oua97] Steve Oualline. *Practical C++ Programming*. O'Reilly & Associates Inc., 1997.
- [Pes00] Gilles Pesant. Scheduling physicians by constraint programming. private communication, 2000.
- [PG99] Gilles Pesant and Michel Gendreau. A constraint programming framework for local search methods. *Journal of Heuristics*, 5:255–279, 1999.
- [Ree93] Colin R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Halsted Press, 1993.
- [Tri98] Geneviève Trilling. Génération automatique d'horaires de médecins de garde pour l'hôpital côte-des-neiges de montréal. *Centre for Research on Transportation (C.R.T.)*, 1998. Travail d'étudiant.
- [WHFP95] G. Weil, K. Heus, P. Francois, and M. Poujade. Constraint programming for nurse scheduling. *Engineering in medicine and biology*, 14(no.4):417–422, 1995.

Appendix

Initial solution is shown in Table 5, while Table 6 shows the initial workload.

Final solution is shown in Table 7, while Table 8 shows the final workload.

In solution tables are *emphasized* the assignments of physicians in their preferred shifts. In workload tables are *emphasized* the workloads which differ from the target.

In both tables of final solution and final workload, the change from initial values are in **bold**.

Monday of week 0 and monday of week 6 are the two holidays of the period.

Initial Solution							
shifts	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
week 0							
am	Boire-Lav.	Rivard	Zinca	Boire-Lav.	Blais	Rivard	Lajeunesse
pm	Boire-Lav.	Belleau	Pelletier	Barsauskas	Champoux	Rivard	Lajeunesse
night	Boire-Lav.	Pitre	Belleau	Barsauskas	Blais	Rivard	Lajeunesse
week 1							
am	Blouin	Pitre	Blais	Rivard	Champoux	Blouin	Blouin
pm	Lajeunesse	Belleau	Pelletier	Barsauskas	Lajeunesse	Champoux	Blouin
night	Rivard	MJK	Blouin	Boire-Lav.	Lajeunesse	Blouin	Blouin
week 2							
am	Boire-Lav.	Belleau	Zinca	Rivard	Blais	Champoux	Bichai
pm	Roy	Pitre	Pelletier	Barsauskas	Lajeunesse	Champoux	Bichai
night	Roy	Pitre	Champoux	Barsauskas	Blais	Champoux	Bichai
week 3							
am	Champoux	Belleau	Lajeunesse	Rebolledo	Champoux	Pitre	Pitre
pm	Boire-Lav.	Pitre	Pelletier	Barsauskas	Blais	Pitre	Pitre
night	Pelletier	Bichai	Belleau	Zinca	Lajeunesse	Pitre	Pitre
week 4							
am	Blouin	Belleau	Zinca	Boire-Lav.	Champoux	Belleau	Boire-Lav.
pm	Rivard	Pitre	Lajeunesse	Barsauskas	Blais	Belleau	Boire-Lav.
night	Rivard	Champoux	Pitre	Barsauskas	Blais	Belleau	Boire-Lav.
week 5							
am	Roy	Belleau	Lajeunesse	Rebolledo	Champoux	Blais	Roy
pm	Pelletier	Pitre	Blais	Barsauskas	Roy	Blais	Roy
night	Roy	MJK	Blouin	Pitre	Lajeunesse	Blais	Roy
week 6							
am	Champoux	Rivard	Zinca	Roy	Blais	Rivard	Roy
pm	Champoux	Belleau	Pelletier	Barsauskas	Champoux	Rivard	Roy
night	Champoux	Pelletier	Belleau	Barsauskas	Blais	Rivard	Roy
week 7							
am	Blouin	Pitre	Lajeunesse	Rivard	Belleau	Barsauskas	Lajeunesse
pm	Boire-Lav.	Roy	Pelletier	Barsauskas	Blais	Barsauskas	Lajeunesse
night	Rivard	Roy	Blouin	Boire-Lav.	Lajeunesse	Barsauskas	Lajeunesse
week 8							
am	Blouin	Champoux	Zinca	Roy	Lajeunesse	Zinca	Zinca
pm	Roy	Pitre	Pelletier	Barsauskas	Roy	Zinca	Zinca
night	MJK	Champoux	Pitre	Barsauskas	Blais	Zinca	Zinca

Initial Solution

shifts	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
week 9							
am	<i>Blouin</i>	<i>Belleau</i>	<i>Pitre</i>	<i>Rebolledo</i>	<i>Champoux</i>	<i>Blais</i>	<i>Bichai</i>
pm	<i>Rivard</i>	<i>Blais</i>	<i>Pelletier</i>	<i>Barsauskas</i>	<i>Lajeunesse</i>	<i>Blais</i>	<i>Bichai</i>
night	<i>Pelletier</i>	<i>Bichai</i>	<i>Belleau</i>	<i>Zinca</i>	<i>Lajeunesse</i>	<i>Blais</i>	<i>Bichai</i>
week 10							
am	<i>Blouin</i>	<i>Champoux</i>	<i>Zinca</i>	<i>Pitre</i>	<i>Belleau</i>	<i>Champoux</i>	<i>Boire-Lav.</i>
pm	<i>Boire-Lav.</i>	<i>Rivard</i>	<i>Pelletier</i>	<i>Barsauskas</i>	<i>Blais</i>	<i>Champoux</i>	<i>Boire-Lav.</i>
night	<i>Rivard</i>	<i>MJK</i>	<i>Blouin</i>	<i>Barsauskas</i>	<i>Blais</i>	<i>Champoux</i>	<i>Boire-Lav.</i>
week 11							
am	<i>Blouin</i>	<i>Lajeunesse</i>	<i>Pitre</i>	<i>Rebolledo</i>	<i>Blais</i>	<i>Pelletier</i>	<i>Pelletier</i>
pm	<i>Boire-Lav.</i>	<i>Champoux</i>	<i>Pelletier</i>	<i>Barsauskas</i>	<i>Belleau</i>	<i>Pelletier</i>	<i>Pelletier</i>
night	<i>Champoux</i>	<i>Boire-Lav.</i>	<i>Pelletier</i>	<i>MJK</i>	<i>Lajeunesse</i>	<i>Pelletier</i>	<i>Pelletier</i>
week 12							
am	<i>Blouin</i>	<i>Lajeunesse</i>	<i>Zinca</i>	<i>Boire-Lav.</i>	<i>Belleau</i>	<i>Blais</i>	<i>Bichai</i>
pm	<i>Champoux</i>	<i>Pitre</i>	<i>Pelletier</i>	<i>Barsauskas</i>	<i>Blais</i>	<i>Blais</i>	<i>Bichai</i>
night	<i>Zinca</i>	<i>Champoux</i>	<i>Belleau</i>	<i>Pelletier</i>	<i>MJK</i>	<i>Blais</i>	<i>Bichai</i>
week 13							
am	<i>Blouin</i>	<i>Champoux</i>	<i>Zinca</i>	<i>Rebolledo</i>	<i>Blais</i>	<i>Blouin</i>	<i>Blouin</i>
pm	<i>Lajeunesse</i>	<i>Rivard</i>	<i>Pitre</i>	<i>Boire-Lav.</i>	<i>Lajeunesse</i>	<i>Blouin</i>	<i>Blouin</i>
night	<i>Boire-Lav.</i>	<i>Pitre</i>	<i>Champoux</i>	<i>Bichai</i>	<i>Pelletier</i>	<i>Blouin</i>	<i>Blouin</i>

Table 5: initial solution

Physician Names	Status (hours/week)	Workload (target)			
		Days	Nights	Weekends	Holidays
Barsauskas	35	13 (14)	6 (6)	1 (3)	0 (1)
Belleau	35	12 (13)	5 (6)	1 (3)	0 (0)
Blais	35	13 (14)	6 (6)	3 (2)	0 (1)
Bichai	21	0 (0)	3 (4)	3 (2)	0 (0)
Blouin	24	9 (9)	4 (3)	4 (2)	0 (0)
Boire-Lav.	24	9 (9)	4 (4)	2 (2)	1 (0)
Champoux	35	13 (14)	6 (6)	2 (2)	1 (0)
Lajeunesse	35	13 (14)	6 (6)	2 (2)	0 (0)
Pelletier	35	12 (13)	6 (6)	2 (2)	0 (0)
Pitre	32	12 (11)	6 (5)	2 (2)	0 (0)
Rebolledo	12	5 (4)	0 (0)	0 (0)	0 (0)
Rivard	24	9 (7)	4 (3)	2 (2)	0 (0)
Roy	35	8 (6)	3 (3)	2 (2)	0 (0)
Zinca	21	8 (8)	3 (4)	2 (2)	0 (0)
MJK	35	0 (0)	6 (6)	0 (0)	0 (0)

Table 6: initial workload

Final Solution

shifts	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
week 0							
am	Barsauskas	Lajeunesse	Zinca	Boire-Lav.	Blais	Rivard	<i>Lajeunesse</i>
pm	Barsauskas	Belleau	<i>Pelletier</i>	<i>Barsauskas</i>	Champoux	Rivard	<i>Lajeunesse</i>
night	Barsauskas	Pitre	<i>Belleau</i>	<i>Barsauskas</i>	Pelletier	Rivard	<i>Lajeunesse</i>
week 1							
am	<i>Blouin</i>	Pitre	Blais	Rivard	Champoux	Barsauskas	Belleau
pm	Lajeunesse	Belleau	<i>Pelletier</i>	<i>Barsauskas</i>	Lajeunesse	Barsauskas	Belleau
night	Champoux	MJK	<i>Blouin</i>	Boire-Lav.	<i>Lajeunesse</i>	Barsauskas	Belleau
week 2							
am	Boire-Lav.	Belleau	Zinca	Rivard	Blais	Champoux	<i>Bichai</i>
pm	Roy	Pitre	<i>Pelletier</i>	<i>Barsauskas</i>	Lajeunesse	Champoux	<i>Bichai</i>
night	Roy	Pitre	Pelletier	<i>Barsauskas</i>	<i>Blais</i>	Champoux	<i>Bichai</i>
week 3							
am	Champoux	Belleau	Lajeunesse	Rebolledo	Champoux	<i>Pitre</i>	<i>Pitre</i>
pm	Boire-Lav.	Pitre	<i>Pelletier</i>	<i>Barsauskas</i>	Blais	<i>Pitre</i>	<i>Pitre</i>
night	MJK	Bichai	<i>Belleau</i>	Zinca	<i>Lajeunesse</i>	<i>Pitre</i>	<i>Pitre</i>
week 4							
am	<i>Blouin</i>	Belleau	Zinca	Boire-Lav.	Champoux	Belleau	<i>Boire-Lav.</i>
pm	Rivard	Pitre	Lajeunesse	<i>Barsauskas</i>	Blais	Belleau	<i>Boire-Lav.</i>
night	<i>Rivard</i>	Champoux	Pitre	<i>Barsauskas</i>	<i>Blais</i>	Belleau	<i>Boire-Lav.</i>
week 5							
am	Roy	Belleau	Lajeunesse	Rebolledo	Champoux	<i>Blais</i>	Roy
pm	<i>Pelletier</i>	Pitre	Blais	<i>Barsauskas</i>	Roy	<i>Blais</i>	Roy
night	Roy	MJK	<i>Blouin</i>	Zinca	<i>Lajeunesse</i>	<i>Blais</i>	Roy
week 6							
am	Blais	Rivard	Zinca	Roy	Blais	Rivard	Roy
pm	Blais	Belleau	<i>Pelletier</i>	<i>Barsauskas</i>	Champoux	Rivard	Roy
night	Blais	Pelletier	<i>Belleau</i>	Champoux	<i>Blais</i>	Rivard	Roy
week 7							
am	<i>Blouin</i>	Pitre	Lajeunesse	Rivard	Belleau	Barsauskas	<i>Lajeunesse</i>
pm	Boire-Lav.	Roy	<i>Pelletier</i>	<i>Barsauskas</i>	Blais	Barsauskas	<i>Lajeunesse</i>
night	<i>Rivard</i>	Roy	<i>Blouin</i>	Boire-Lav.	<i>Lajeunesse</i>	Barsauskas	<i>Lajeunesse</i>
week 8							
am	<i>Blouin</i>	Champoux	Zinca	Blais	Lajeunesse	Zinca	Zinca
pm	Roy	Pitre	<i>Pelletier</i>	<i>Barsauskas</i>	Champoux	Zinca	Zinca
night	MJK	Champoux	Pitre	<i>Barsauskas</i>	<i>Blais</i>	Zinca	Zinca
week 9							
am	<i>Blouin</i>	Belleau	Pitre	Rebolledo	Champoux	Belleau	<i>Bichai</i>
pm	Rivard	Blais	<i>Pelletier</i>	<i>Barsauskas</i>	Lajeunesse	Belleau	<i>Bichai</i>
night	Pelletier	Bichai	<i>Belleau</i>	Zinca	<i>Lajeunesse</i>	Belleau	<i>Bichai</i>
week 10							
am	<i>Blouin</i>	Champoux	Zinca	Pitre	Belleau	Champoux	<i>Boire-Lav.</i>
pm	Boire-Lav.	Rivard	<i>Pelletier</i>	<i>Barsauskas</i>	Blais	Champoux	<i>Boire-Lav.</i>
night	<i>Rivard</i>	MJK	<i>Blouin</i>	<i>Barsauskas</i>	<i>Blais</i>	Champoux	<i>Boire-Lav.</i>
week 11							
am	<i>Blouin</i>	Lajeunesse	Pitre	Rebolledo	Blais	Pelletier	Pelletier
pm	Boire-Lav.	Champoux	<i>Pelletier</i>	<i>Barsauskas</i>	Belleau	Pelletier	Pelletier
night	Champoux	Boire-Lav.	Pelletier	Bichai	<i>Lajeunesse</i>	Pelletier	Pelletier
week 12							
am	<i>Blouin</i>	Lajeunesse	Zinca	Boire-Lav.	Belleau	<i>Blais</i>	Barsauskas
pm	Champoux	Pitre	<i>Pelletier</i>	<i>Barsauskas</i>	Blais	<i>Blais</i>	Barsauskas
night	Zinca	Blais	<i>Belleau</i>	<i>Barsauskas</i>	MJK	<i>Blais</i>	Barsauskas
week 13							
am	<i>Blouin</i>	Champoux	Zinca	Belleau	Blais	Blouin	Blouin

Final Solution

shifts	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
pm	Lajeunesse	Barsauskas	Pelletier	Boire-Lav.	Lajeunesse	Blouin	Blouin
night	Boire-Lav.	Pitre	Champoux	Bichai	Pelletier	Blouin	Blouin

Table 7: final solution

Physician Names	Status (hours/week)	Workload (target)			
		Days	Nights	Weekends	Holidays
Barsauskas	35	14 (14)	6 (6)	3 (3)	1 (1)
Belleau	35	13 (13)	5 (6)	3 (3)	0 (0)
Blais	35	14 (14)	6 (6)	2 (2)	1 (1)
Bichai	21	0 (0)	4 (4)	2 (2)	0 (0)
Blouin	24	9 (9)	4 (3)	2 (2)	0 (0)
Boire-Lav.	24	9 (9)	4 (4)	2 (2)	0 (0)
Champoux	35	14 (14)	6 (6)	2 (2)	0 (0)
Lajeunesse	35	14 (14)	6 (6)	2 (2)	0 (0)
Pelletier	35	13 (13)	6 (6)	2 (2)	0 (0)
Pitre	32	11 (11)	5 (5)	2 (2)	0 (0)
Rebolledo	12	4 (4)	0 (0)	0 (0)	0 (0)
Rivard	24	7 (7)	3 (3)	2 (2)	0 (0)
Roy	35	6 (6)	3 (3)	2 (2)	0 (0)
Zinca	21	8 (8)	4 (4)	2 (2)	0 (0)
MJK	35	0 (0)	6 (6)	0 (0)	0 (0)

Table 8: final workload

Acknowledgements

I would like to thank all the staff of the Centre for Research on Transportation (C.R.T.), starting from Director Michel Gendreau, and then Louis-Martin Rousseau, the technical support, and the secretaries.

I would thank in particular Prof. Gilles Pesant for his useful help, advice, corrections and suggestions, and for being always very obliging.